US005790664A

United States Patent [19]

Coley et al.

[11] Patent Number: 5,790,664

[45] Date of Patent: Aug. 4, 1998

[54] **AUTOMATED SYSTEM FOR MANAGEMENT OF LICENSED SOFTWARE**

[75] Inventors: **Christopher D. Coley**, Morgan Hill; **Ralph E. Wesinger, Jr.**, Livermore, both of Calif.

[73] Assignee: **Network Engineering Software, Inc.**, San Jose, Calif.

[21] Appl. No.: **607,081**

[22] Filed: **Feb. 26, 1996**

[51] **Int. Cl.**[6] **H04L 9/00**

[52] **U.S. Cl.** **380/4**

[58] **Field of Search** 380/4, 25; 395/712

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,888,798	12/1989	Earnest	380/4
5,023,907	6/1991	Johnson et al.	380/4
5,138,712	8/1992	Corbin	395/712
5,204,897	4/1993	Wyman	380/4
5,260,999	11/1993	Wyman	380/4
5,375,206	12/1994	Howter et al.	395/712
5,438,508	8/1995	Wyman	380/4
5,495,411	2/1996	Ananda	380/4
5,548,645	8/1996	Ananda	380/4
5,579,222	11/1996	Bains et al.	395/712
5,629,980	5/1997	Stefik et al.	380/4

OTHER PUBLICATIONS

Élan Computer Group, Inc., "*Élan License Manager Technical Overview*", pp. 1–10, (1995).

GLOBEtrotter Software, Inc., "*FLEXlm End User Manual*", Chapters 1–7, (1996).

Primary Examiner—Salvatore Cangialosi

Attorney, Agent, or Firm—McDonnell Boehnen Hulbert & Berghoff

[57] **ABSTRACT**

Methods and apparatuses are disclosed for providing a system for automatically tracking use of a software and also for determining whether the software is validly licensed and enabling or disabling the software accordingly. Exemplary systems involve attaching a licensing system module to a software application. Records of valid licenses are stored in the database maintained by the software provider. The licensing system module transparently forms a license record inquiry message. The message is transparently sent to the database over a public network, such as the Internet, to determine whether a valid license record exists in the database for the software application. The database forms and returns an appropriate response message that is interpreted by the licensing system module. The software application can then be appropriately enabled or disabled by the licensing system module. The receipt of the license record inquiry can be recorded in the database to monitor software use.

25 Claims, 7 Drawing Sheets

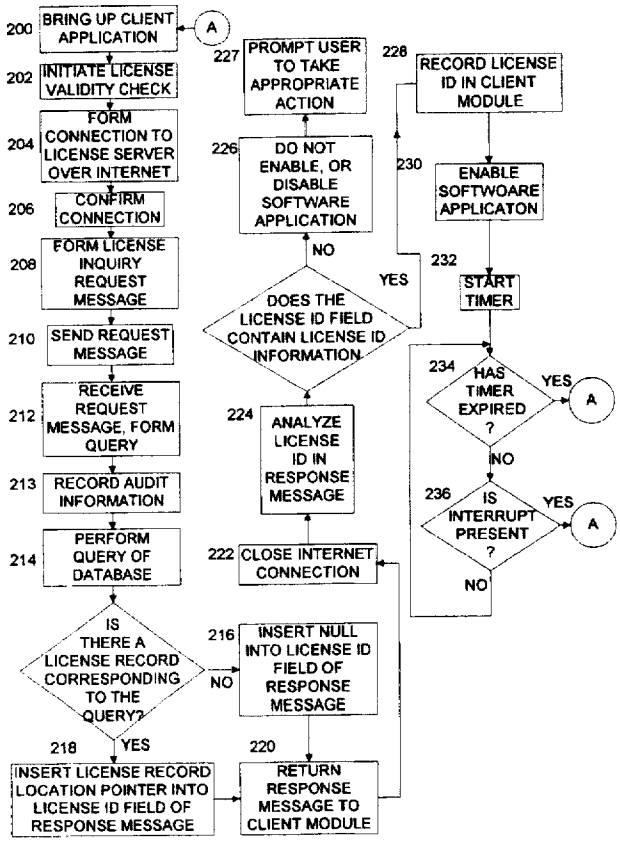
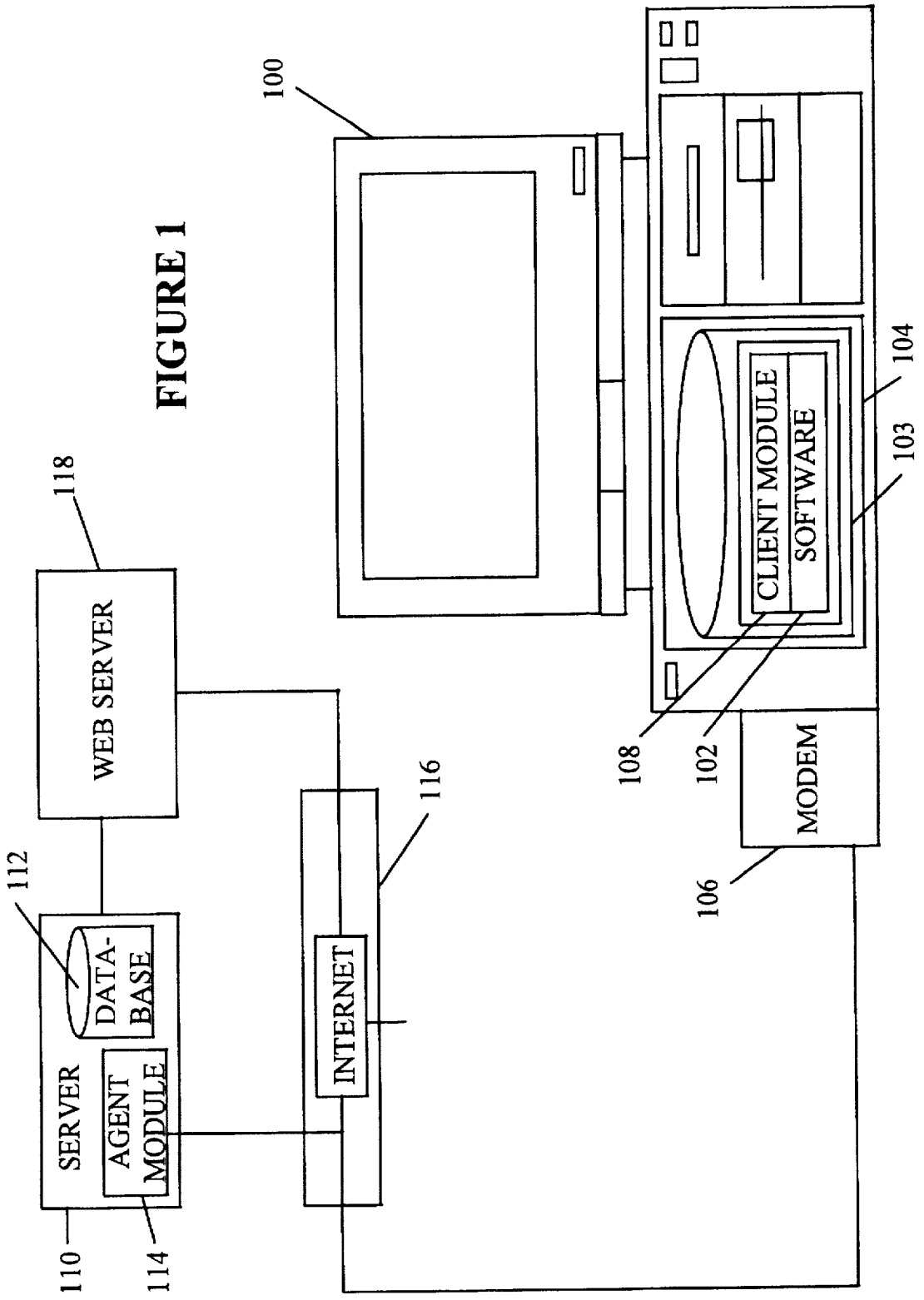


FIGURE 1


100
104
103
CLIENT MODULE
SOFTWARE
108
102
MODEM
106
116
INTERNET
118
WEB SERVER
112
DATA-BASE
SERVER
AGENT MODULE
110
114

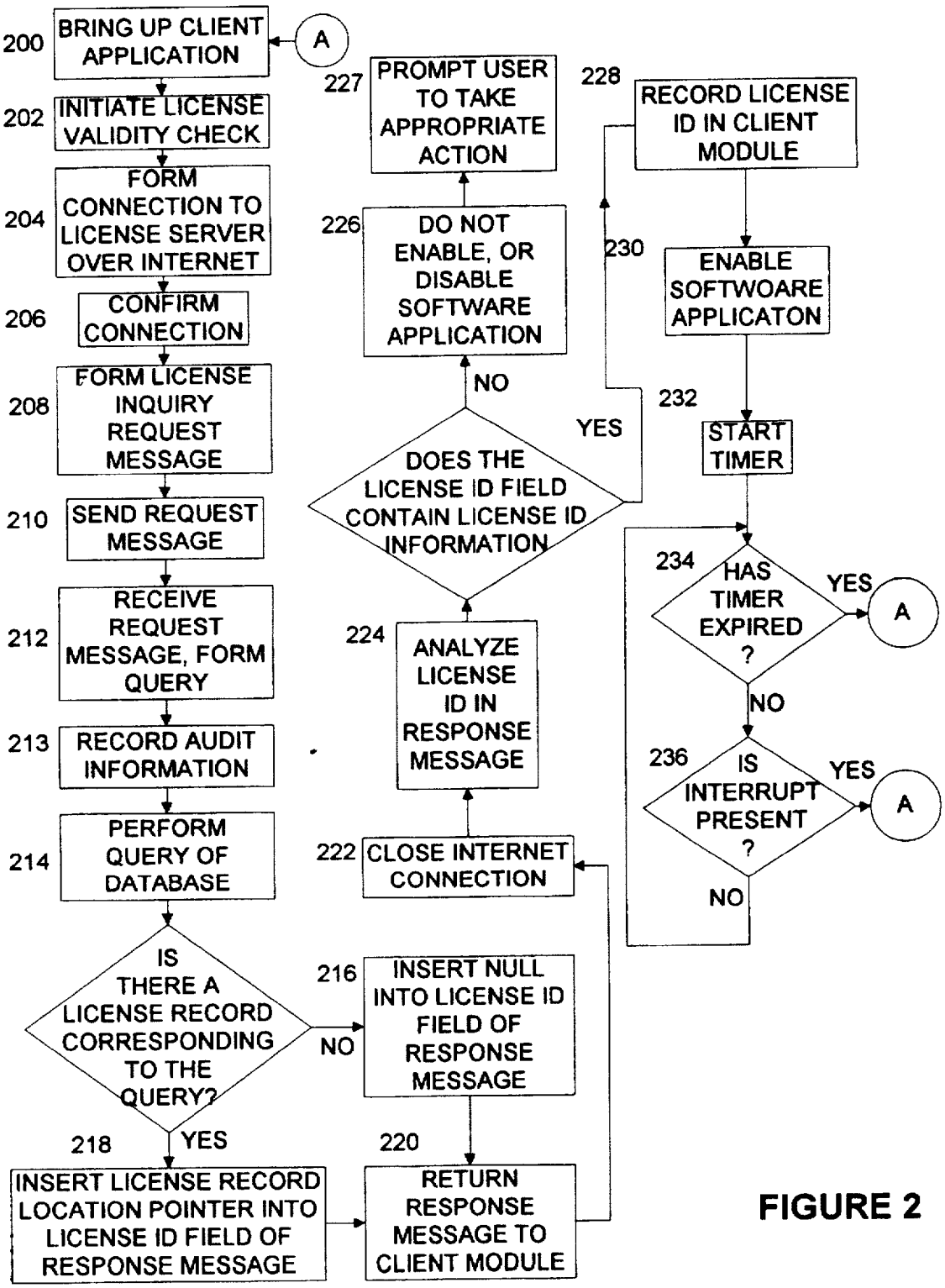

200
BRING UP CLIENT APPLICATION
A
202
INITIATE LICENSE VALIDITY CHECK
204
FORM CONNECTION TO LICENSE SERVER OVER INTERNET
206
CONFIRM CONNECTION
208
FORM LICENSE INQUIRY REQUEST MESSAGE
210
SEND REQUEST MESSAGE
212
RECEIVE REQUEST MESSAGE, FORM QUERY
213
RECORD AUDIT INFORMATION
214
PERFORM QUERY OF DATABASE
IS THERE A LICENSE RECORD CORRESPONDING TO THE QUERY?
NO
216
INSERT NULL INTO LICENSE ID FIELD OF RESPONSE MESSAGE
YES
218
INSERT LICENSE RECORD LOCATION POINTER INTO LICENSE ID FIELD OF RESPONSE MESSAGE
220
RETURN RESPONSE MESSAGE TO CLIENT MODULE
222
CLOSE INTERNET CONNECTION
224
ANALYZE LICENSE ID IN RESPONSE MESSAGE
DOES THE LICENSE ID FIELD CONTAIN LICENSE ID INFORMATION
NO
226
DO NOT ENABLE, OR DISABLE SOFTWARE APPLICATION
227
PROMPT USER TO TAKE APPROPRIATE ACTION
YES
228
RECORD LICENSE ID IN CLIENT MODULE
230
ENABLE SOFTWOARE APPLICATON
232
START TIMER
234
HAS TIMER EXPIRED ?
YES
A
NO
236
IS INTERRUPT PRESENT ?
YES
A
NO

FIGURE 2

FIGURE 3

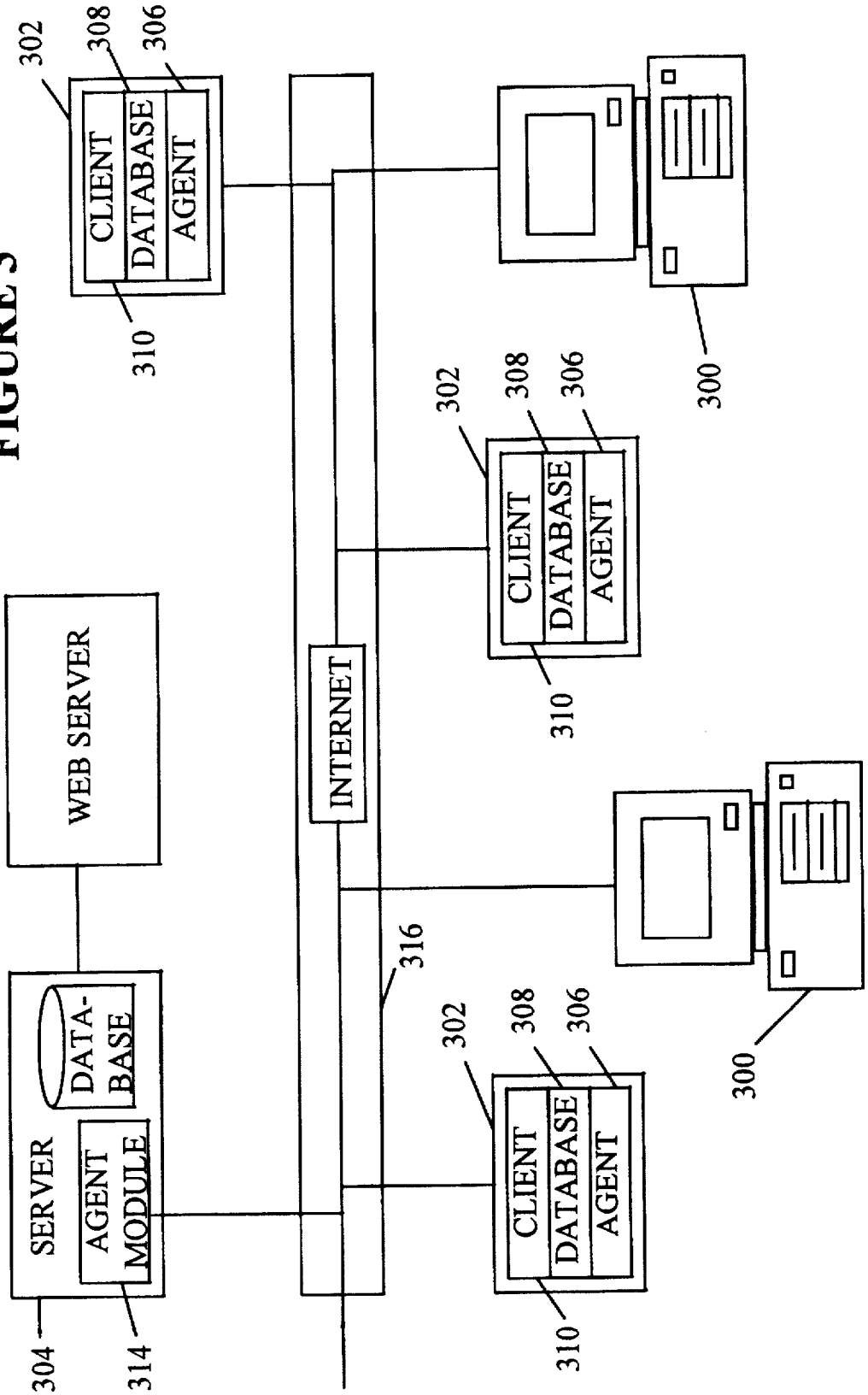
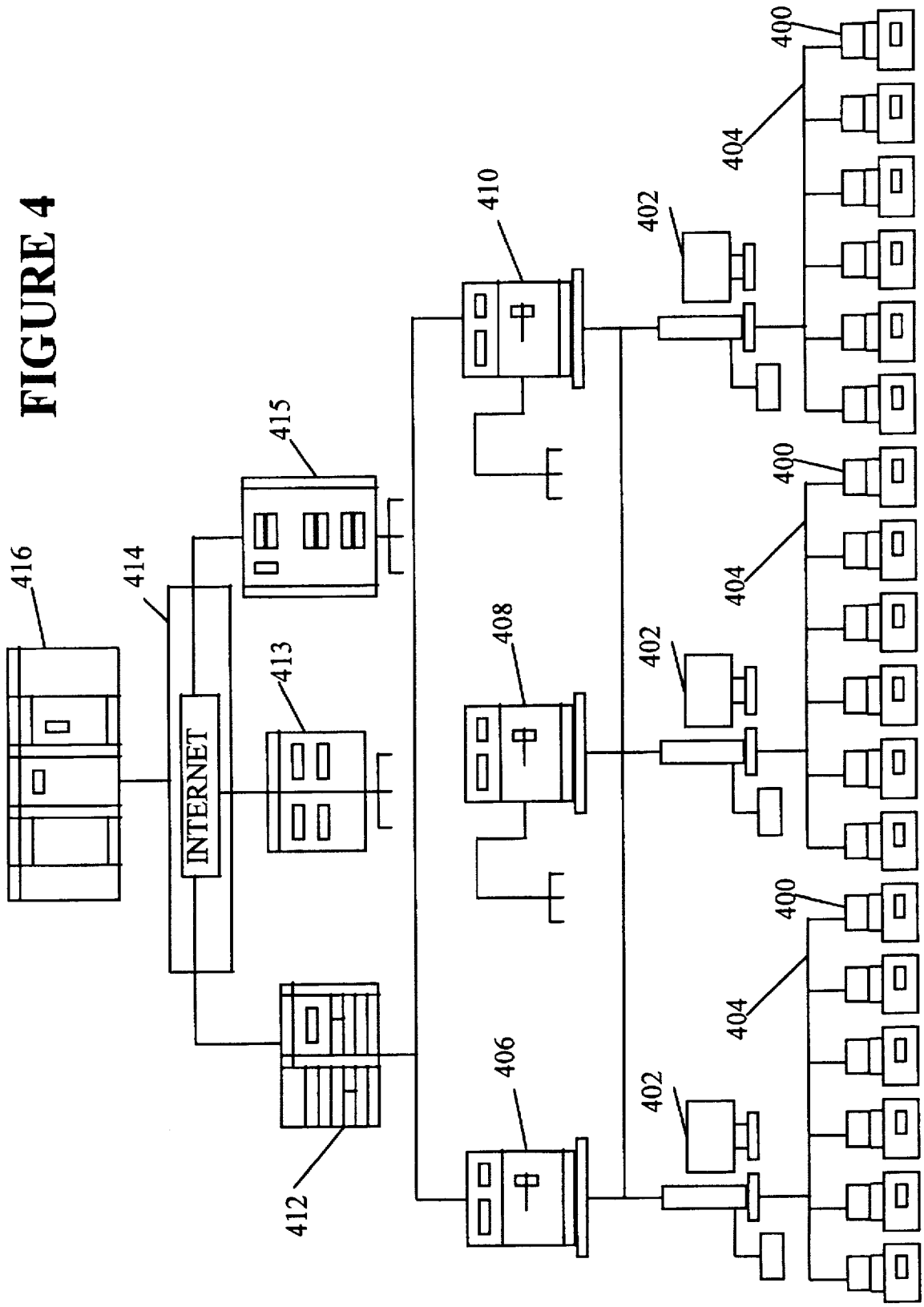

302
308
306
CLIENT
DATABASE
AGENT
310
300
WEB SERVER
INTERNET
316
SERVER
DATA-BASE
AGENT MODULE
304
314

FIGURE 4



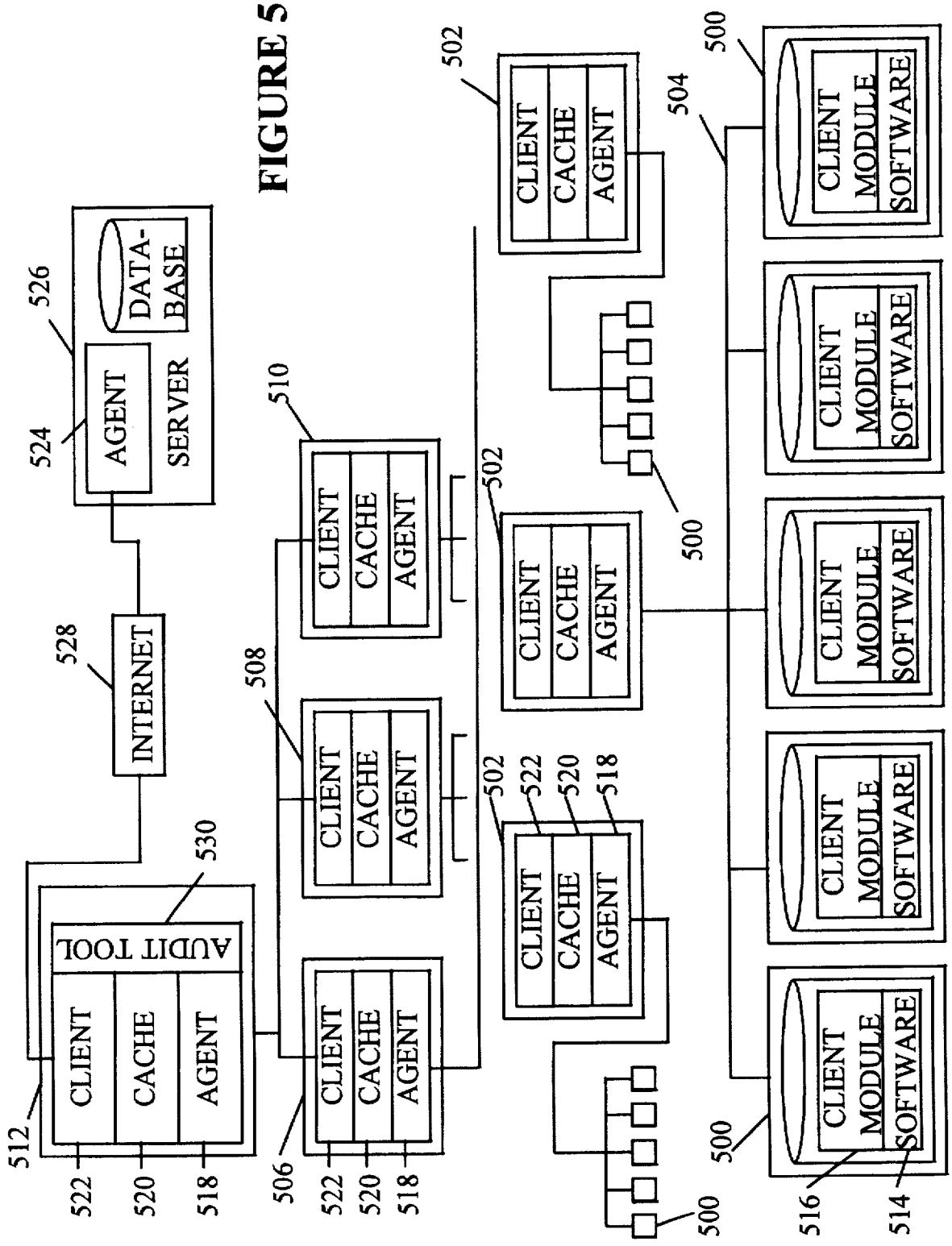

526
DATA-BASE
524
AGENT
SERVER
528
INTERNET
530
AUDIT TOOL
CLIENT
CACHE
AGENT
512
522
520
518
506
508
510
502
504
500
516
514
CLIENT MODULE SOFTWARE

FIGURE 5

FIGURE 6

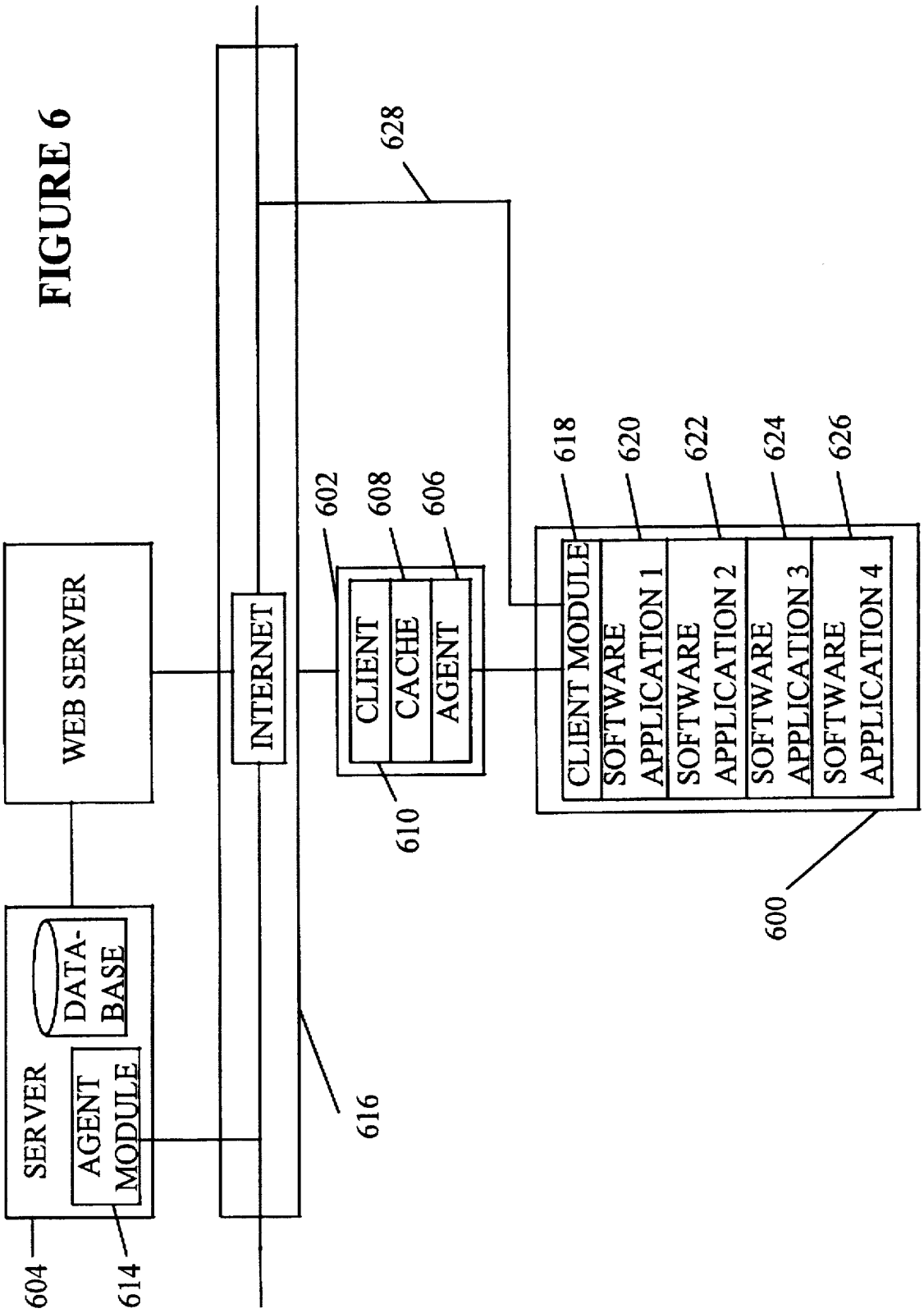
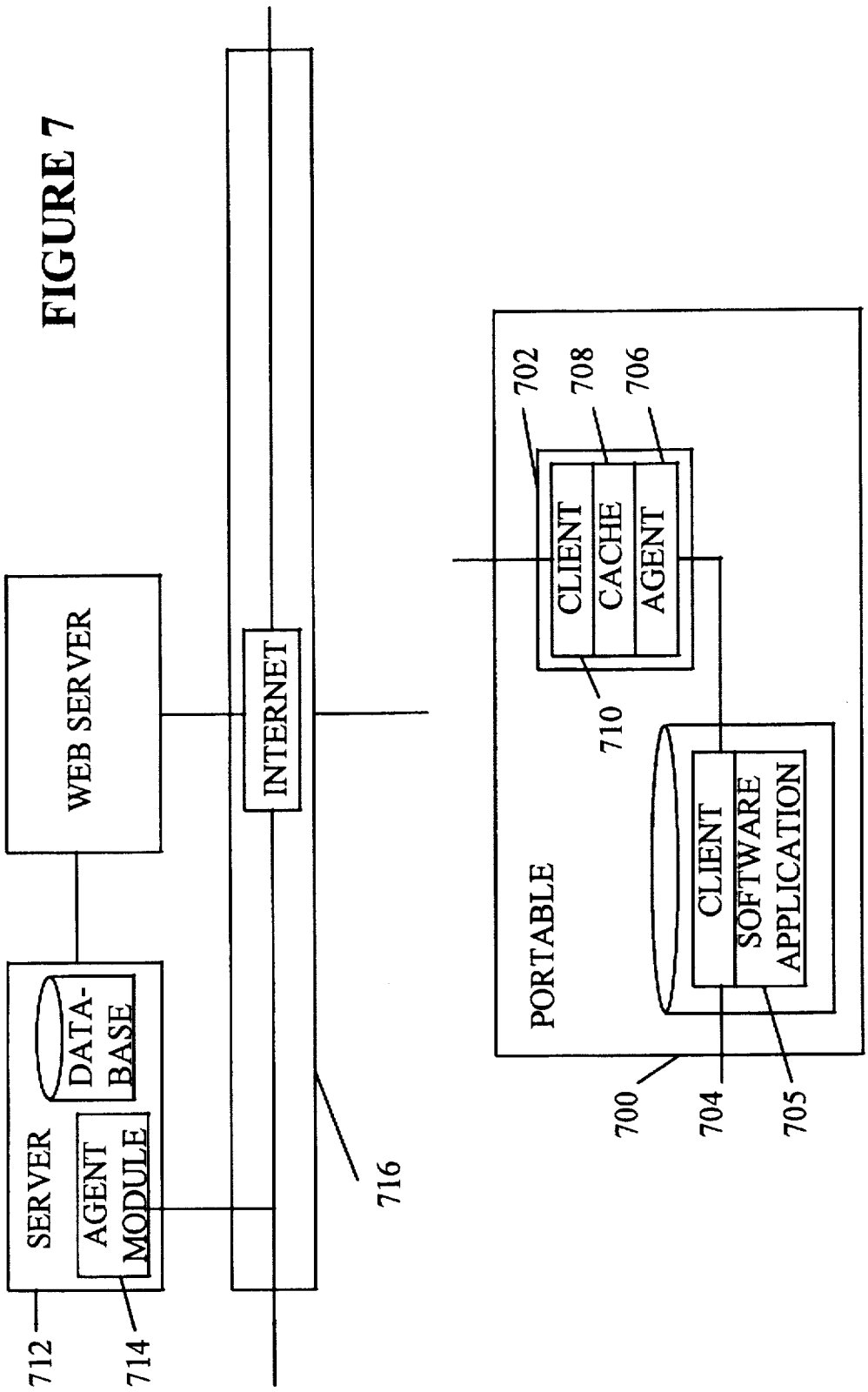

WEB SERVER
INTERNET
SERVER
AGENT MODULE
DATA-BASE
CLIENT
CACHE
AGENT
CLIENT MODULE
SOFTWARE APPLICATION 1
SOFTWARE APPLICATION 2
SOFTWARE APPLICATION 3
SOFTWARE APPLICATION 4
600
602
604
606
608
610
614
616
618
620
622
624
626
628

FIGURE 7


WEB SERVER
INTERNET
SERVER
AGENT MODULE
DATA-BASE
712
714
716
PORTABLE
CLIENT CACHE AGENT
CLIENT SOFTWARE APPLICATION
702
708
706
710
700
704
705

AUTOMATED SYSTEM FOR MANAGEMENT OF LICENSED SOFTWARE

BACKGROUND

The present invention relates to software licensing, and in particular to a system for automated monitoring and management of licensed software.

It is well known that software is not purchased, but only licensed for use. Software, unlike manufactured products, can be freely copied and distributed. Hence, software providers are largely limited in their choice of control means. Unfortunately, a software license is merely a legal mechanism, and can not literally prevent illicit copying of proprietary software. A typical software license grants a permit to use the licensed software application on a particular machine and, perhaps, the generation of backup copies for personal use. A software license provides a software provider with a legal instrument against impermissible use of licensed software. However there remains no effective mechanism for preventing or monitoring illicit copying or illegal proliferation in the first place. Hence, software providers must rely on the public to not pirate software, and rely on their licensees to abstain from furnishing copies of software to friends or others. A significant amount of software piracy occurs in commercial settings. Commercial licensees are usually vigilant about license compliance. However, even the most attentive MIS manager (Management Information Systems) cannot prevent employees from copying software off of company machines for their personal use. As a result of illicit copying, software providers must adjust their prices, forcing legitimate buyers to pay higher prices to offset revenue losses.

Although the estimates for losses due to piracy are estimated in the billions of U.S. dollars, such estimates remain mere projections. This is because there way of determining how widely software is copied. More specifically, there is no mechanism by which to monitor the proliferation and use of software, copied or otherwise. Statistics regarding the use of legitimately purchased software also remains unknown to software providers and vendors. Despite sales data, purchased software may be found ineffective by users and sit, unused, on hard drives. Consequently, software providers may not have a firm understanding of how their products are being received by users. Whether used or not, most proprietary software contains some security mechanism and/or registration mechanism.

There are a number of schemes designed to prevent software from being copied, or to make use of copied software unduly burdensome. These schemes, however, are largely ineffective, complex, and add to development costs. Furthermore, for every protection scheme devised by programmers, there are hackers who will diligently go about undermining them. A first line of defense is to encourage legitimate users to register their licensed software.

Registration of software provides a software provider with a record of a valid license. Registration typically involves filling out and mailing a registration card that is provided in an off-the-shelf software package. A user may be asked to write in the serial number of the software set, along with other pertinent information. The defense mechanism in registration, albeit weak, is that a software provider will only render assistance and support to properly registered users. That is, a software provider will refuse to grant assistance to a user unless the user has properly registered their software.

The registration process also may involve responding to prompts generated by the software when it is first run. The prompt may be a security system asking the user to enter the serial number and/or a codeword to enable the software. The codeword may be a word appearing at prompt-designated locations in a user manual. This security scheme operates on the premise that a pirate will not ordinarily have a copy of the user manual. After the software is up and running, it may periodically prompt the user to re-enable the software by entering a different codeword appearing at varying locations in the user manual. This scheme is subverted by copying the manual and registration number.

An on screen registration/enablement process may involve writing the registration number to disk. This is only possible with floppy disks as CD-ROMS are, at present, a largely read-only medium. If the disk is used again to load the software application, the software may prompt the user with a warning that the software has already been loaded (e.g., "IS LOADING OF THIS SOFTWARE PERMITTED? RECORDS INDICATE THAT THIS SOFTWARE HAS ALREADY BEEN LOADED. YOU MAY BE IN VIOLATION OF YOUR LICENSE AGREEMENT."). However, reloading of software may be normal in the event of hard disk failure. Consequently, software providers cannot feasibly prevent the software application from being loaded more than once. Furthermore, if first run registration enablement is required, copiers can simply copy the software repeatedly prior to registering the original copy.

Another security technique is to enable a software application for a defined period of time. This usually involves incorporating a date/time checking mechanism into the software application. Such a mechanism may be used where a software provider wants to supply, for example, a 30-day demonstration version of a software application for user evaluation. If the user decides to purchase a license following the evaluation period, the user may contact the software provider and supply payment information. Following approval (e.g., credit card) or receipt (i.e., check) of the payment, the software provider may supply the user with a regular copy of the software, or provide instructions or a codeword to disarm or reset the date/time checking mechanism.

In operation, a date/time checking mechanism records a date/time stamp when a software application is first brought up. Alternatively, or in addition, the date/time mechanism may start a timer when the application is brought up. The date/time stamp is compared with the system date/time information maintained by the computer to determine if the software application is to be disabled. To subvert such a system, users have been known to reset the system date and system time to prevent expiration. In response, some software providers have resorted to writing complex code schemes to disable the software in the event that the system date is tampered with. Such a security mechanism is often used to control licensed software used in a commercial setting.

Software sold for use in a commercial or institutional setting is frequently licensed for a predefined period of time. When such software is used on desktop computers, such computers are typically networked. The networked computers are usually connected to a file server, which file server may itself be tended by a computer management system that monitors and controls various file server groups. The file server computers act as a central location at which the desktop computers in the file server group can access files and applications. The file server also may facilitate the control of licensed software on the desktop computers. This occurs in the situation where the commercial software license is a so-called "floating license."

Commercial software licenses for operating a plurality of desktop computers normally are of two varieties: "fixed" or "floating." A fixed license permits a software application to run on certain designated computers (e.g., computer numbers one through five, in a ten computer file server group, are designated for the licensed software application). A floating license permits a certain number of applications to run on any number of computers at a given time. So an application operating under a floating license may be allowed to simultaneously run on no more than ten of twenty computers in a network at any given time. Licensing management software is maintained in the network file server to monitor the number of floating licenses being used.

Commercial software is prone to installation interruptions as it almost always requires involved enablement procedures. In accordance with regular industry practices, commercial software applications are ordinarily enabled following their installation by contacting the software provider for enablement instructions and/or enabling codes. This process is rarely instantaneous. The software provider usually confirms that the software license is proper and paid for before faxing, e-mailing, or even using regular mail, to provide a set of enabling instructions, enabling codes, or disk(s) with which to bring the application up. Consequently, the software remains disabled until additional instructions are supplied and followed, which are usually sent only after an enablement request is approved.

Management of floating licenses on networked computers involves two control software components: an application portion, and an authenticator portion. The application portion is nested within an application running on a desktop computer. The authentication portion is a code module contained in the file server that monitors and authorizes applications running on the desktop computers. When a user attempts to open the application software, the application portion code communicates with the authenticator code module to check to see if a floating license is available. If the maximum number of floating licenses are already being used, the software application is not allowed to open. Licensing control software also may be used to monitor defined term licenses to disable software in networked machines after license expiration.

If a commercial license expires, the software may be disabled, midstream, preventing users from completing projects. Re-enablement requires contacting the software provider to purchase an additional license or extension. This may require re-execution of enablement procedures with new instructions or codes. Hence, it may take some time before the software application is up and running again, which situation can seriously inconvenience users.

The common shortcoming shared by all licensed software, is that it requires some form of manual intervention for registration, enablement, and/or re-enablement. Manual intervention is cumbersome and can render software useless until it is enabled or re-enabled. The paramount issue is, however, that software providers have no mechanism for monitoring and controlling the actual use, whether legitimate or illicit, of their product. Proprietary software is misappropriated on a global scale causing massive losses to software providers, which losses are inevitably passed on to legitimate licensees.

What is needed is a licensing system that allows software use to be monitored in an automated fashion, without user input. Moreover, a software licensing system is needed that permits a software provider to transparently control the use of licensed software.

SUMMARY

The present invention addresses the foregoing problems by providing a system for automatically determining whether a software application is licensed. In accordance with the invention, a generic licensing module, or "client module," is provided that a software provider can attach to a software application. A software application having a client module attached thereto is hereinafter referred to as a "client application." In accordance with preferred embodiments of the invention, a client application loaded on a computer having access to a public network, such as the Internet, automatically reports to a computer maintained by a software provider. The client module is a program, application, or like composition of code that is preferably nested in a compiled version of a software application (i.e., to form a client application). However, the client module can, in alternative embodiments of the invention, be attached to a previously compiled software application. Whether it is referring to a program nested in, or attached to a software application, the term client module is used throughout the present disclosure.

A client module utilizes the public network as a means to transparently send license inquiry request messages to, and receive license inquiry response messages from, a license server maintained by a software provider. The license server has a database on which license information, or records, are stored. The license server also can record information contained in license inquiry request messages, and thereby audit use of client applications. The license record can identify a license in accordance with a hardware address, or hardware identifier of the computer, such as an IP address.

Operation of an exemplary system incorporating the invention involves the client module in a client application generating inquiries that are sent to the license server in the context of an Internet communication session. The license server responds to the inquiry by investigating its database to determine whether a corresponding license record is present. The license server then forms an appropriate response message that is sent back to the client module.

If it is determined that the client application is not licensed (i.e., the database does not contain a corresponding license record), the response sent by the licensing server does not allow the client application to be enabled. If the client application is licensed (i.e., the database contains a record of a license), the response can allow the client application to be enabled, or re-enabled. In sum, the client application must be enabled for it to properly operate. Communication between the computer and the licensing server is generally transparent to a user. That is, the client module automatically forms a connection with the license server, sends a message, and receives a response, all without user input or notification.

Alternatively, an exemplary embodiment of the invention can be used to monitor use of client applications. Operation of an exemplary system incorporating the invention for monitoring client application use involves using the client module in the client application to generate messages that are sent to the licensing server. Such messages can be sent over any public network to which a user computer, upon which the client application is loaded, is connected. For example, a message can be sent to the license server in the context of an Internet communication session. The license server tracks, or audits, the use of client applications by recording pertinent information contained in a message generated by a client module. A database can be used to store the information. A software provider or vendor can access recorded information stored in the database to generate

client application use reports. Such an auditing system can be a part, or a function of, a system for enabling, validating and/or disabling licensed software (i.e., client applications). When configured as such, audit data can be derived from license inquiry request messages. Furthermore, the license record database in the license server can be used to store the collected audit data. Alternatively, a separate database can be used.

In a personal computer setting, an exemplary process in accordance with the invention may involve utilizing a modem, or like device, in the computer. The client module generates and sends a license validity inquiry request message to a regional or central license server maintained by the software provider. The license server contains an agent module for communicating with the client module and a database containing license records. The database in the license server is checked to see if a valid license record exists for the requesting client application and computer. If so, a message is transmitted back that allows enablement or re-enablement of the client application. The licensing server also can record information corresponding to the request in the database containing the license records, or in a different database.

If a license record is not found, the client application is not enabled. A menu can be presented asking whether the user would like to purchase a license, and thus enable the software. The menu may direct a user to a Web homepage where a license can be purchased, automatically open a session to such a homepage, or provide a telephone number of a sales representative or automated operator. Optionally, the user can initiate a demonstration mode of operation to evaluate the client application.

In a commercial or institutional computer environment (i.e., networked computers), a licensing system in accordance with exemplary embodiments of the invention can involve a hierarchical arrangement of licensing modules arranged between client application(s) and a license server. At the desktop computer level, a client module monitors one, or more, software applications on the desktop computer. Computers, such as file servers, residing at each level of the network, between the desktop computers and the license server, contain licensing modules. Licensing modules include an agent component for communicating with a downstream client, a cache component for interim storage of license information, and a client component for communicating with an upstream agent.

Operation in an exemplary process involves the client module in a desktop computer communicating upstream with an agent component in a licensing module. A client component in that licensing module communicates upstream with an agent component in a next licensing module, whose client component, in turn, communicates with a next upstream agent, and so on. This arrangement is continued upward to converge on a license server which contains an agent module. However, the license server is maintained by the software provider. Consequently, the uppermost licensing module in the institutional network communicates with the license server by initiating a connection over a public network, such as the Internet. License enablement information is supplied to the upper-most licensing module by the license server, which information is propagated back downstream via the licensing modules. The cache components in the licensing modules can be used to store license records so that license inquiries can be addressed without having to forward the validation inquiry request messages to the license server.

In a preferred embodiment of the invention, as applied in a network environment, the client and agent elements are generic. That is, client modules in the desktop computers and client components in each of the network level computers are substantially similar. Agent components in the licensing modules and the agent module in the license server also are substantially similar. The cache components of the licensing modules are used to store information that defines the structure of license records. Any client can communicate with any agent and vice versa. This arrangement facilitates network configuration flexibility.

Operation of an exemplary system incorporating the invention in the network environment involves a client module in a client application forming and communicating a license validity inquiry request message upstream. An agent component in a nearest upstream licensing module receives the request. The licensing module checks its cache to determine if a license record exists corresponding to the request. The licensing module can then respond with an appropriate message. If the license information is not found at that level, the licensing module can forward the request upstream to determine if the license is of record in an upstream cache. This may continue up to the license server. When and if a license record is found, the information is copied into the cache of the licensing module nearest to the computer originating the request for future reference. In accordance with another aspect of systems incorporating the invention, the licensing modules periodically communicate upstream to fetch license information in order to revise and maintain currency of their cached license records. By doing so, a validity request can be addressed by a nearest upstream agent. Furthermore, periodic checking can permit management of request message traffic on the license server.

In accordance with an exemplary embodiment of the invention, license information is organized in class and sub-class designations. License information maintained on the license server covers blocks of underlying computers rather than the individual desktop computers themselves. This provides for efficient communication of license information between the license server, licensing modules, and desktop computers.

Whether used in a personal or commercial computing environment, systems incorporating the present invention allow client software to be enabled or re-enabled at any time without significant delay. Software applications operating in accordance with the present invention can be installed on any computer in the world having access to a public network, such as the Internet. If so desired, a client application can be configured to not operate unless it receives acknowledgment of the presence of a valid license record. As the use of computers expands globally, a licensing system in accordance with the present invention can ensure that a client application operating on any computer in the world is properly licensed.

In addition, in a personal or network computing environment, systems incorporating the present invention can be used to monitor client application use. Such a system can operate by recording information from license validity inquiry requests messages that are received at the license server. Client application use, and corresponding details, can be recorded when the client module sends license inquiry request messages back to the license server. Alternatively, the system can be used in an audit-only mode whereby the client applications report relevant information back to the license server, but do not require an enabling response message to continue operating. As the use of computers expands globally, a licensing system in accordance with the present invention can assist in tracking client application use and proliferation.

Preferred embodiments of the present invention take advantage of the fact that an increasing number of computers, and computer networks, have direct access to the Internet. Systems in accordance with the present invention can utilize the Internet as the medium over which license validity inquiry request inquiries and their corresponding responses are transmitted.

In accordance with exemplary embodiments of the invention, if a client application does not receive enablement information, the client application is not enabled, or is disabled. Hence, any software application that contains a client module accordance with the invention, can be automatically enabled, or disabled. Furthermore, use of client module equipped applications can be tracked. Such a system allows software to be freely distributed while ensuring that a license is taken for its use, or at the very least, ensuring that the use of the software can be tracked.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing, and other objects, features and advantages of the present invention will be more readily understood upon reading the following detailed description in conjunction with the drawings in which:

FIG. **1** depicts a desktop computer in accordance with an exemplary embodiment of the invention;

FIG. **2** depicts a flow diagram for operating the arrangement of FIG. **1** in accordance with an exemplary process incorporating the present invention;

FIG. **3** depicts a regional server scenario in accordance with an exemplary embodiment of the invention;

FIG. **4** depicts a computer network arrangement in accordance with an exemplary embodiment of the invention;

FIG. **5** depicts a symbolic representation of the computer network of FIG. **4**;

FIG. **6** depicts an additional exemplary embodiment of the invention wherein multiple software application licenses are managed; and

FIG. **7** depicts an additional exemplary embodiment of the invention in a portable computer.

DETAILED DESCRIPTION

FIG. **1** depicts a personal computer system in accordance with an exemplary embodiment of the invention. The system includes a personal computer **100** that has a client application **103** residing on a hard drive **104**. The client application **103** is comprised of a software application **102** and a client module **108**. The computer **100** includes a modem **106**. The client module **108** operates to enable or disable the software application **102** pursuant to a response from a license server **110** in the context of license validity inquiries. The license server **110** contains a database **112** having license records recorded thereon, and an agent module **114** that communicates with the client module **108**. The licensing server **110** is typically maintained by the software provider who developed the software application **102**. Alternatively, the license server **110** can be maintained by a contracted service provider. In a preferred embodiment, the client module **108** and the agent module **114** communicate over the Internet **116**. However, the client and agent can communicate over any public network. As used herein, the term public network encompasses not only networks that are freely available to the public, generally, but also any private network which can be subscribed to. The depiction of the client module **108** is merely for descriptive and illustrative purposes. The client module **108** can be code nested within the software application **102**.

In accordance with an exemplary embodiment of the invention, the client module **108** automatically initiates a process to determine whether the software application **102** is validly licensed. This can happen each time the client application **103** is brought up. The licensing module **108** operates transparently and utilizes the modem **106** to form a connection with the licensing server **110**. Once the connection is made, the client module **108** sends a license validity inquiry request message to the license server **110**. The agent module **114** receives the request and queries the database **112** to determine whether a license record exists that corresponds to the client application **103** and computer **100**. The license server **110** also can record relevant information contained in the license validity inquiry request message to audit the use of client application.

Pursuant to the query, a response message is generated and returned back to the client module **108** by the agent module **114**. If the query finds a valid license record, the response message indicates the license record's existence and location in the database **112**. The client module **108** can record the license record location for future reference. If a license record is not located in the database **112**, an appropriate response message is generated and returned. The presence of a record permits the client module **108** to enable, or to allow continued operation of (i.e., re-enable), the software application **102**.

If the query of the database **112** returns an indication that a license does not exist, the client module **108** can pursue any one of many courses of action. The first possible course of action is to not enable the software application **102**, or to it if it is presently operating. The client module **108** also may supply an appropriate message to the user indicating that a license does not exist for the client application. Alternatively, the client module **108** can supply a message instructing the user as to where and how a license can be purchased. This may involve directing the user to telephone the software provider's sales department, or the message can direct the user to an appropriate Web site homepage on a Web server **118** where the software can be purchased.

If a license is purchased via a homepage on the Web server **118**, the Web server **118** can automatically update the database **112** with the information supplied by the user or the computer **100** (e.g., IP address supplied in the context of a Web session), thereby creating a valid license record. A subsequent client module license inquiry will allow the client application **103** to be enabled. If the license purchase transaction fails for any reason (e.g., a credit card supplied is not approved), the license record can be removed from the database **112** by the software provider. This has the effect of disabling the client application in a next license validity inquiry. The practical effect of such a system is to be able to provide software that is freely distributable. Any user in possession of the client application **103** will not be able to use it until a license is procured and a license record is established.

An exemplary process of operating the arrangement depicted in FIG. **1** is shown in the flowchart of FIG. **2**. The frequency and timing for performing a license validation check can be selected according to the discretion of the software application designer. In the exemplary process depicted, the client module performs an initial check each time the software application is brought up (step **200**). The license validity checking process is initiated (step **202**) by utilizing a modem to form an Internet connection between the computer **100** and a licensing server (step **204**). This may be done by having the client module instruct the modem to dial a 1-800 number maintained by the software provider that accesses a local Internet gateway (if used in the United States).

Once the connection is confirmed (step **206**), the client module **103** forms a license validity inquiry request message (step **208**). The request message may contain information such as the application name, the application version number, a date/time stamp, the name of a license server **110** (if several license servers are maintained by the software provider), and a hardware identifier, such as the IP address of the computer **100**. After formation, the request message is sent to the license server **110** (step **210**) over a public network*. The agent module **114** in the license server **110** forms a query (step **212**) to determine whether a corresponding license record is stored in the database **112** (step **214**). The agent module **114** also can record audit information from the request message (step **213**). If the query locates a record of a license for the request, a response message is returned having a license ID field comprising a pointer to the location of the license record in the database **112** (step **218**). If the query does not locate a record of a license for the request, a response message is returned having a null indication in the license ID field (step **216**). The response message is returned to the client module **108** (step **220**) after which the Internet connection is closed (step **222**).

The client module **108** investigates the response message to determine whether the license ID field contains a license ID (step **224**). If the license ID field is null, the client module **108** fails to enable the software application, or disables it (step **226**). The client module **108** may then prompt the user with any variety of messages (step **227**). For example, the user may be prompted to assess whether a demonstration period of operation would be acceptable. If so, this information can be recorded in the client module **108** and be passed upstream in the context of a next validity inquiry request message. The server **110** will record this information in the database **112**. Alternatively, the user can be prompted to contact a sales representative or automated operator to purchase a license, or directed to a Web homepage where a license for the software application can be purchased. In the event of a license purchase, the database **112** can be automatically updated to record the license. Thereafter, a validity check will find a license record and allow the client application **103** to be enabled.

If the license ID field contains a license ID, this information is recorded by the client module **103** for future use (step **228**). The client module **108** then enables the software application **102** (step **230**). The client module may, at this point, start a timer (step **232**) for periodic checking of license validity. Such a validity check is automatically initiated when the timer expires (step **234**). The client module also can be configured to initiate a validity check whenever an interrupt is present indicating a certain activity (step **236**), such as printing or saving.

Periodic checks performed at timer expiration, or upon appropriate interrupt, use the license ID, which is a pointer, to directly access the database record corresponding to the license. If the license record is found, a response message indicates so, the software remains enabled, and the timer is reset. If the record is found empty, it may indicate that the license has expired. The response message will indicate this, and the software can be disabled. Alternatively, the user may be requested to renew the license within a certain period of time before the software application **102** is disabled.

The date/time stamp information passed upstream in the license validity inquiry request message can be used to detect whether the system date/time information on the computer **100** has been tampered with. This is done by comparing the date/time information passed in the request message with the date/time information maintained on the licensing server **110**. Furthermore, in preferred embodiments of the invention, license ID information is communicated between the client module **108** and the agent module **114** in an encrypted form, as explained in greater detail below.

In the event that no license is found, several response options are available which vary according to the requirements of, and discretion of a designer of the software application **102**. As previously mentioned, a response can be to provide the user with a phone number through which a software license can be purchased, or to direct the computer user to a Web homepage maintained by the software provider. Alternatively, the client module **108** can directly initiate a session with the Web server **118** that supports a homepage through which the user can purchase a license. A first screen on such a homepage can prompt the user to indicate whether the purchase of a full license would be desirable, or whether a demonstration period is preferable to evaluate the application. If neither of these options is selected the session is terminated. If the user opts to take a license, the user can be prompted with questions asking which features in the software application are to be enabled (the price of the license can be adjusted accordingly). The session can conclude with the presentation of a payment screen inviting the user to enter credit card information, or to call a sales representative in order to supply payment information.

If credit card information is supplied in the homepage session, it can be gathered using the system disclosed in the U.S. Pat. application Ser. No. (BDSM Attorney Docket No. 025553-014) entitled: "System for Securely Storing Information Received Over a Public Network," by Coley and Wesinger, filed on Feb. 6, 1996, and incorporated herein by reference in its entirety. Once the credit card information is entered, a response message can be sent to the client module **108** temporarily enabling the software application **102**. The database **112** can then automatically updated with a license record. If a credit card turns out to be invalid, the license server database **112** can be updated accordingly by removing the license record and thereby disabling the software pursuant to a next inquiry.

The exemplary inventive system described above allows client applications (i.e., software application having client modules) to be freely distributed while reasonably ensuring that they are, or will be, licensed if used. Any software application having a licensing system client module attached will not operate unless and until the license system client module receives authority to enable the software application. Such a system allows global proliferation of the software, even in the form of a copy. However, such widespread use of client applications may result in the license server **110** being inundated with validity request message traffic. A dedicated license server can be set up to handle all of the license inquiry traffic for a particular software application. Alternatively, some form of traffic management can be invoked.

Traffic management can take many forms. It can involve establishing regional license servers according to a geographic arrangement that permits efficient response to any licensing inquiry request messages. A client application initialization process can be used wherein a user enters the location (e.g., zip code, city, and country). This information can be used by the client module to select an appropriate autodial telephone number whereby a nearest software provider license server can be accessed.

Regional license servers can receive license information propagated from a central licensing server. Client modules

can contact their regional license server to assess whether license records exist for their corresponding client applications. Because use of the Internet is contemplated as a means for communicating licensing inquiry and response messages, the regional license server designated for a particular client module can be assigned in accordance with efficient network, and/or geographic, considerations. This can facilitate provision of a swift response to license validity inquiry request messages.

An arrangement of regional licensing servers in accordance with an exemplary embodiment of the invention is depicted in FIG. **3**. Regional licensing servers **302** are comprised of an agent component **306**, a database component **308**, and a client component **310**. A client module in a desktop machine **300** communicates with an agent component **306** in an assigned regional license server **302** over a public access network, such as the Internet **316**. The client component **310** in the regional license servers **302** communicates with an agent module **314** in a central license server **304**. License information can be systematically directed from the central license server **316** to appropriate regional license servers **302** in accordance with information supplied when the license is procured. Alternatively, the regional license servers **302** can systematically request license record information with which to update their database components **308**.

A variation on the regional license servers **302** can be used in a licensing management system incorporating the invention for application in a commercial setting. That is, wherein computers are networked in a hierarchical arrangement within a company or institution.

FIG. **4** depicts a commercial network system in accordance with an exemplary embodiment of the invention. Desktop machines **400** are organized in file server groups. The file server groups are administered by file server computers **402** through networks **404**. The file server groups can, for example, serve various design teams in a research and development facility of a corporation. The file servers **402** in the R&D facility are, in turn, tended by a minicomputer **406**. The minicomputer **406**, and minicomputers **408** and **410** at other facilities (e.g., manufacturing and sales) are networked under a main computer **412** located, e.g., at the headquarters of the corporation. In accordance with an embodiment of the invention, each desktop computer **400** contains a client module for monitoring one or more client applications. The client modules in the desktop computers **400** communicate upstream with licensing modules contained respective file server computers **402**. The licensing modules in the file server computers **402** communicate with a licensing module in the minicomputer **406**, which licensing module, in turn, communicates with a licensing module in the main computer **412** at the corporation headquarters. The licensing module in the main computer **412** uses a public network, such as the Internet **414**, to communicate with a license server **416** maintained by a software provider who developed the software application(s) on the desktop computers **400**. Main computers **413** and **415** at other corporations or institutions also can communicate with the license server **416** to communicate license inquiry and response messages.

A representation of the network scenario depicted in FIG. **4**, illustrating licensing system components in accordance with an exemplary embodiment of the invention, is shown in FIG. **5**. Various network computers are depicted in symbolic form to assist in illustrating the components involved in the exemplary embodiment of the invention. Desktop computers **500** contain software applications **514** having licensing system client modules **516** attached thereto. The desktop computers **500** are tended by group file servers **502** on networks **504**. Each of the group file server computers **502**, minicomputers **506**, **508** and **510**, and a main computer **512** contain a licensing module. A licensing module comprises an agent component **518**, a cache memory component **520**, and a client component **522**. The license server **526** maintained by the software provider contains an agent module **524**. For any of the licensing modules in the intermediate computers between the desktop computer **500** and the license server **526**, the licensing module's client component **522** communicates with the agent component **518** of an upstream licensing module, or with the agent module **524** of the license server. The licensing module's agent component **518** communicates with a downstream licensing module's client component **522**, or a client module **516** in a desktop computer **500**. Communication between the upper-most licensing module in the internal network (i.e., licensing module **512**) and the agent module **524** in the license server **526**, is conducted over a public network, such as the Internet **528**.

An audit function can be implemented in a networked embodiment of the present invention in a number of ways. For example, the upper-most licensing module **512** can maintain software, such as an audit tool **530**, that tracks use of client applications in underlying computers in the network. An audit report can be periodically generated and sent upstream to the license server **526**. The license server **526** can record and interpret the audit report to monitor use of client application software. Alternatively, license validity inquiry request message traffic from individual client applications can be recorded in the license server **526**. Audit information can be used to generate billing invoices.

An additional aspect of the aforementioned audit system permits an MIS manager at a corporation or institution to monitor the use of client applications for internal audit purposes. Such a system operates by monitoring license inquiry traffic passing through a network to and from a license server. In an exemplary embodiment, such a system involves maintaining internal auditing software (e.g., a tool or utility program) in an upper-most level licensing module in an internal network. A report can be generated by the internal auditing software tool. Data in the report can be derived from information collected at the upper-most licensing module. The MIS manager can use the internal audit reports to manage the licensing arrangements of the client applications on the network. For example, if a network of twenty desktop computers is frequently using a maximum number of floating licenses for a particular client application, the MIS manager can ascertain this by reviewing internal audit records, and take appropriate action.

In accordance with preferred embodiments of the invention in a network setting, each of the agent-type components, and each of the client-type components are generic. That is, any given agent component **518**, and the agent module **524** in the license server **526**, is substantially similar. The same is true of the client components **522** and the client modules in the desktop computers **500**. The parameters maintained by, and passed between various license system elements defines the licensing system structure.

In accordance with a preferred embodiment of the invention, the license system operates by distributing licensing information to the cache components **520** in the licensing modules in response to inquiry requests. The information contained in a particular cache component **520** is specific to subordinate software applications **514**, or licensing modules. In accordance with a preferred embodiment, license information is organized by class designations. Individual

licenses for client applications on desktop machines **500** can be covered by sub-class licenses maintained in the cache components **520** in the file server computers **502**. The sub-class licenses on the file server machines **502** can, in turn, fall under a class license maintained in the cache component **520** of the minicomputer **506**. The class license maintained on the minicomputer's licensing module can be designated under a block license maintained in a cache component **520** of the main computer **512**. The client component **522** of the main computer's licensing module communicates with the license server **526** to verify block licenses.

Validity inquiry traffic is managed in the exemplary system of FIG. **5** through the use of the aforementioned license class designation structure and through periodic updating of caches in licensing modules. By maintaining license information in a class/sub-class designation, a single license validity inquiry generated by, for example, a file server's licensing module, can enable a sub-class license that covers the client applications of all of the desktop computers **500** in the file server's group. This is generally more efficient than having each desktop computer **500** individually validate its own license through the license server **526**. Furthermore, because the sub-class license maintained on the file server itself can fall under a class license maintained on the minicomputer, the validity inquiry by the file server may be addressed by the minicomputer rather than forwarding the request further upstream.

In accordance with a preferred embodiment of the invention, licensing modules periodically and systematically initiate license validity inquiries upstream. The responses to the periodic inquiries are recorded into the cache components of the licensing modules. As a result, client modules and client components need ordinarily communicate with a next upstream agent to be enabled, re-enabled, or to update their caches. Periodic self-validation updating of the licensing modules also provides for more uniform request traffic on the licensing server **526**. This can provide efficiencies in propagating license information from the licensing server in response to license validity inquiries. Of course, the foregoing system can operate by passing individual license validity request and response messages through the licensing modules.

The result of the hierarchical arrangement is to control the flow of validity inquiry traffic received and responded to by the licensing server. If a license server dedicated to a particular client application is employed, the license server may be able to handle all of the license inquiry traffic received. However, if a single license server handles license validity inquiry requests for a variety of client applications, request traffic management may be needed. By arranging the system in a hierarchical fashion and designating the licenses in accordance with class and hierarchical subclass designations, the licensing server can manage request traffic for a several different client applications.

Client components and modules contain a set of license management procedures for handling license record information. In accordance with an exemplary embodiment of the invention, three validation procedures are used: Check Out License, Validate License, and Check In License. Check Out License is responsible for the initial enablement of a client application. Validate License is responsible for periodic re-enablement of a client application. Check In License can be used for decrementing a floating license count, or indicating client application status in a database license record when use of a client application is completed. Check In License also may be used for disabling a license or feature after its use in the event that there is a per-use cost associated with the software. The following exemplary process descriptions are provided for a case wherein a client application is validating itself directly, or indirectly (i.e., through one or more licensing modules) over the Internet.

Each of the procedures involves initially forming a connection with a corresponding upstream agent. The agent can be the agent component of a licensing module in an upstream network file server, or like machine containing a licensing module. Or the upstream agent may be the agent module in a license server in which case the connection is formed over a public network, such as the Internet.

When a client application is first brought up, the Check Out License procedure is initiated. The purpose of the Check Out License procedure is to enable the software application to which the client module is attached. In addition, the Check Out License procedure can be used by to track the proliferation of a client application. In accordance with an exemplary process incorporating the invention, the client module's Check Out License call generates a client data structure containing: the name of the software application, any feature name(s) that is to be enabled, the name of the upstream agent component/module, a hardware identifier of the computer containing the client application, a date/time stamp, and a version number of the licensing system. Other fields are created in the client data structure that are filled in by the upstream licensing module or license server.

Prior to, concurrently, or following the formation of the client data structure, the client module forms a connection with the upstream agent component designated in the client data structure. Once the connection is completed, the Check Out License procedure sends a license validity inquiry request message to the upstream agent, which request message contains the contents of the client data structure. Alternatively, the entire data structure can be sent. The information from the client data structure is used by the upstream agent to form a query with which to determine the existence, if any, of a license record for the client application. In the case where the upstream agent is the agent module in the license server, the query is performed on the server database. In the case where the upstream agent is an agent component in a licensing module (e.g., in a file server) the query is performed on the licensing module's cache component.

If the query finds a license record in the database or cache, the location of that record, in the form of a pointer, is generated for storage in a license ID field in the client data structure. The licensing module also generates an authorization ID for storage in a field of the same name in the client data structure. If the query does not locate a license record, the license ID field and authorization ID field are left blank (i.e., nulled). Following the query, a license validity inquiry response message is formed by the agent and returned to the client. The response message contains the license and authorization IDs, if any, that are to be stored in the client data structure.

The client module investigates the client data structure content returned by the agent component/module to analyze the license and authorization ID information contained therein, if any. If these fields indicate the presence of a valid license, the client module enables the software application. The license and authorization IDs are stored in the client data structure in the client module for future license validation checks. If the data structure fields for the license and authorization IDs are null, the client application is not enabled and the client data structure is deleted from the client module.

The database or cache query also may involve a comparison of the date/time stamp contained in the client data structure with date/time information maintained by the license server or license module system. This added security measure can detect a user's tampering with system time and date information on their computer. If the particular software application is operating in a demonstration mode for a predefined period of time, the date/time stamp passed in the client data structure can be used as an initial check of whether the demonstration period has expired. If there is date/time corruption, the client application can be disabled.

In an audit function, the contents of a message generated by the Check Out License procedure can be recorded in the license server or upstream licensing module. Relevant information recorded by the license server can include, but is not limited to, the hardware identifier of the computer upon which the client application is loaded (e.g., the IP address), the application name, feature names, and the application version number. Additionally, a client module registration number, or like identifying means, can be nested in a client application. This registration number also can be passed upstream in a request message for audit recordation. A client module registration number allows a software provider to track the proliferation paths and patterns of copied client applications. Consequently, a software provider can determine the source of copied software. A software provider can force a user to attach a non-connected computer to a network access point by supplying a prompt indicating to the user that a client application will not enable unless and until it checks in. Consequently, if the user chooses to use the client application (i.e., attach the computer to a network access point), the software provider is informed of the client application's use.

A second process initiated by a client module or a client component is the Validate License procedure. This process determines whether a previously enabled client application is still validly licensed. The procedure can be called at any time. For instance, the initiation of the Validate License procedure can be in response to a timer expiration (i.e., a periodic check) or at the appearance of a system interrupt (e.g., printing is selected). The Validate License procedure, in accordance with an exemplary process incorporating the invention, generates a new client data structure containing: the name of the software application, any feature name(s) that is to be enabled, the name of the upstream agent component/module, the license ID, the authorization ID, a date/time stamp, and the version number of the licensing system.

Following completion of the upstream connection, the Validate License procedure sends a license validity inquiry request message to the upstream agent, which request message contains the new client data structure contents. The license ID information in the client data structure is used by the upstream agent to directly access the memory location where the license record was previously located. If the query finds a license record, a new authorization ID is generated by the agent and stored in the authorization ID field in the client data structure. If a license record no longer exists, the license ID field and authorization ID field are nulled. A license validity inquiry response message containing the client data structure is then formed by the agent and returned to the client module.

The client module investigates the data structure contents returned in the response message to analyze license and authorization ID information. If the authorization ID indicates the continued presence of a valid license, the client module allows the client application to remain enabled. The new authorization ID is stored in the client data structure in the client module for future license validation checks or to check in the license. If the data structure fields for the license and authorization IDs are null, the client application is disabled and the client data structure is deleted. Alternatively, the user can be prompted to indicate whether a new license would be desirable. If so, any of the procedures previously described can be used to procure a new license (e.g., access a Web homepage).

The Check In License procedure can be used to return a license or disable a feature when a user has completed use of the client application or a feature contained therein. The procedure, in accordance with an exemplary process incorporating the invention, generates a new client data structure containing: the name of the software application, any feature name(s) that is to be disabled, the name of the upstream agent component/module, the license ID, the authorization ID, a date/time stamp, and the version number of the licensing system.

Following completion of the upstream connection, the Check In License procedure sends a license check in request message to the upstream agent, which request message contains the new client data structure contents. The license ID information in the client data structure is used by the upstream agent to directly access the memory location where the license record was previously located. The existing license record is modified to indicate the disablement of a feature, or is deleted (if the agent is the agent module in the license server). If the agent is an agent component in a licensing module, the license is designated for deletion. This information is passed upstream in a license check in request message subsequently sent up stream by the licensing module in its next periodic self-validation. The license and authorization ID fields of the client data structure are nulled, and a license check in response message containing the client data structure is then formed by the agent and returned to the client module. The client module then deletes the client data structure.

It is important to note that, in a network setting incorporating the aforementioned exemplary processes, the client components of the licensing modules also can use the Check Out, Check In, and Validate License procedures to update licensing records stored in their cache components. Performance of these procedures by the licensing modules is substantially similar to performance of the procedures by a client module. However, the corresponding license ID, instead of referring to an individual client application, can refer to a class or sub-class license that covers a block of underlying client applications, or licensing modules. As previously mentioned, the Validate License procedure can be periodically initiated by the licensing modules to systematically update the contents of their caches and to provide for request traffic management received at the license server. Automatically auditing cached license records improves the likelihood of a client application, or client component, finding a current license record in a next upstream agent in the context of a Check Out or Validate License procedure.

A licensing system in accordance with the invention, in a preferred embodiment, involves inserting licensing system code into a pre-compiled version of a software application and then compiling that application into a single executable client application. However, in accordance with another embodiment of the invention, the licensing system can be provided as a module that is inserted into to an existing software structure on a computer network. Such a system can be used to monitor software application use in a com-

puter network that does not otherwise have a means to audit application use. This type of system can be used by individuals, such as MIS managers, who wish to audit software application use activity in a network. In an exemplary embodiment, a client module installed in an individual desktop computer "wraps around" software applications selected for monitoring. When the wrapped application used, or seeks a floating license from a file server, a licensing module installed in the file server computer records the activity. In a floating license system, the licensing module can be configured to always reserve a license for use by certain individual computers (e.g., the CEO's computer). Audit records generated by the licensing module can be periodically checked by the MIS manager to see if adjustments are needed (e.g., purchase a greater number of floating licenses).

A variation of the latter system also can be installed in existing network to retrofit a licensing management system. Client modules wrap previously installed applications thereby converting them into client applications. The client modules on individual computers monitor and/or control client application use. The client modules can report to licensing modules in upstream file servers, or report directly to a license server over a public network. Such a system can be used by software providers as an aftermarket component installed on top of existing software systems. In such a setting, a client module can be responsible for handling license validation of more than one client application loaded on the computer. If two or more client software applications on a computer are by a same software provider, the client module can generate a single validity request message covering each wrapped client application. Such a system has the effect of providing a generic licensing validation system for all of the licensed software on a desktop machine supplied by a particular software provider.

A exemplary licensing system that can be retrofitted in an existing computer network is depicted in FIG. **6**. An individual computer **600** has a client module **618** installed therein. The client module **618** is wrapped around one or more software applications **620**, **622**, **624** and **626** to create client applications. In a preferred embodiment, the client applications are specific to the software provider who is retrofitting their networked software with the licensing system. However, if a license record database is configured as a "clearing house," whereby a multitude of software providers consolidate license information in a single server or a network of license servers, the client module **618** can validate software licenses by a variety of software providers. The client module **618** can enable, or validate, the client applications by communicating with a licensing module in a file server **602**. The licensing module in the file server **602** also is a retrofitted component in the license system. Licensing modules are installed in each of the computers (e.g., file servers, minicomputer, main computers) that form the network hierarchy. Operation of such a system is substantially similar to that of a network embodiment of a license management system described above. Alternatively, client modules can communicate directly with a license server **604** over a public network, such as the Internet **616**.

As mentioned in the Background above, software licensing management systems conventionally maintain proprietary, that is application specific, licensing code in software applications (i.e., application portion). Corresponding proprietary licensing code also is maintained in the file server or like network element (i.e., authenticator portion). Consequently, a conventional system typically has to maintain several separate licensing validation programs on a file server to check each of the software applications loaded on sub-tended computers. A system in accordance with the latter embodiment of the invention provides for a generic solution whereby a single client module is maintained on the desktop computer that handles all of the licensing management for the computer's client applications. Hence, the computers and corresponding servers need not contain several licensing system applications each having proprietary code.

Because the exemplary embodiments described above describe use of the Internet as a communications medium, the hardware identifier maintained in the client data structure is preferably the IP address of a computer. However, any other suitable hardware identifier can be used to identify the computer on which a client application or licensing module is loaded. For example, Intel's Pentium®processor contains an internal serial number that can be used as a hardware identifier. Alternatively, a hardware module can be provided that is designed specifically for a licensing system in accordance with the present invention. Such a hardware module can be attached to a parallel port on a computer can be used as a tag to identify the computer. The client module in the computer scans the parallel port to acquire the computer's hardware ID for insertion into a client data structure.

The license ID maintained in the client data structure is used as a pointer to a location in the cache or database of a corresponding upstream licensing agent or server. The pointer designates the location of the licensing record. A license record at any given level, can represent a relationship between the agent and an underlying client. That relationship, as designated by the license ID can be used as a universal designator to replace designating a particular client/agent pair by application name, feature name, IP address, agent name, etc. Use of a license ID provides a more efficient means with which to track and communicate information regarding a particular license.

The authorization ID is used as a means to indicate the status of a license in response to an enablement or validation inquiry. The authorization ID also can be used as an acknowledgement to a Check In License procedure that permits a client application to delete a client data structure. Furthermore, the authorization ID also can be used to return a status message containing an error code, an indication that the server or agent is presently too busy to handle a query, an indication of a connection or communication failure, or any other like message. Another important task facilitated by the authorization ID is its use in supporting encrypted communication between a client and an agent.

Communication between clients and agents, in accordance with preferred embodiments of the invention, is encrypted. In an encryption scheme according to an exemplary embodiment, the authorization ID is used to pass back a key for use in encryption processing. The encryption scheme involves maintaining identical encryption engines in corresponding clients and agents. The key passed by the authorization ID is used as a common seed, or initialization vector, for initializing the respective encryption engines for encryption and decryption of messages. A new key is generated and passed back each time an agent responds to a client inquiry, which key is used in a next round of communications. The key is stored in the client module in the context of storing the client data structure (containing the authorization ID). In a subsequent client-to-agent communication the key is used for client encryption of a message. The message is then decrypted at the agent using the common key.

In order to allow encryption synchronization, all of the data structure fields in an initial communication between a

client/agent pair are unencrypted. Subsequent communications have certain fields encrypted. In response to an initial communication, the agent passes back a first key. The message is then re-sent having designated fields of the client data structure encrypted. The response passed back by the agent in a next communication contains a new key for use in a subsequent communication. In this way, the client and agent maintain encryption synchronization by supplying the same key to their respective encryption engines. In accordance with a preferred embodiment, the key is a random number generated by an agent.

The encrypted fields of the data structure contents passing between a client and agent include the license and authorization Ids, and any proprietary data required for validation, such as floating license information. All of the fields of the data structure contents can be encoded, however, the application name, feature name, hardware identifier, and a licensing system version number are preferably left unencoded. In the event that there is lack in synchronization, a client or agent can look to the unencoded information and revert back to a most recent encryption key. If there is still lack of synchronization, a request is passed that the client return to the initial message state, thereby allowing both the client and agent encryption engines to reset.

The frequency of validation checks is application dependent. A software designer can select when and how often validation checks are to occur, if at all. The licensing system can be configured in accordance with the needs of a particular application. The software license can be validated, or enabled, each time the application is brought up on a computer, or each time a particular feature is used (e.g., printing). The software license also can be validated in response to the expiration of a timer (i.e., periodic validation). If the response to the validation check is returns a null license ID, the client application is disabled. To prevent a user from tampering with a client application, or a licensing module associated therewith, in an effort to disable validation checking, any number of watchdog timers can be nested in the client application. The nested watchdog timer can be used to periodically self-check the client application to determine whether it has been validated within the watchdog period. If so, the watchdog timer is reset. If not, a validation check can be initiated or the client application can be disabled.

Establishing a database license record in the licensing server can be performed in a variety of ways. Software can be purchased and paid for in an interactive commercial transaction conducted over the Internet, as described above. The result of such a transaction is to establish a license record in the licensing server database. A subsequent validation check by the client application will allow the software to be enabled. The database entry also can be formed by pre-authorization. If the software is purchased from a vendor, the vendor, in the context of the transaction can perform the database entry shortly after the software is supplied to the user or company. When the software is brought up on the client computer, an initial validity check will return an enablement response because a license record has already been established. Alternatively, software can be pre-enabled with a temporary term license thereby providing a software provider with a time window in which to establish a license record. Other techniques for establishing an entry in the database, and thus enabling the corresponding client application, include using automated telephone operator systems. A client can call a telephone number and use a touch-tone phone to respond to prompts presented by an automated operator. Hence, any mechanism for initializing the database, and consequently automatically enabling the software, is deemed suitable.

Another aspect of systems operating in accordance with the invention is feature enablement. The systems described above can be used to enable and disable particular features in a client application. Such a situation may occur wherein a software application has several levels of operating capability. For instance, a user can selectively enhance operating capability by selecting features defined in a software feature application menu. In response, an associated client module can invoke the Check Out License procedure wherein the desired feature name is passed upstream. The license, of course, does not exist yet, but the system can be configured to direct the user to, or provide the user with, a feature enablement menu that requests that the user enter credit card information, as described above. Alternatively, the system can be organized to automatically initiate a process that creates a license when new software is brought up. This can involve a mechanism that forms a Web server connection and supplies an authorization message that creates a database license entry. For a commercial client, a software provider can monitor the activation and use of client application features and bill the client accordingly.

As previously discussed, systems in accordance with the present invention can be used to audit the use and proliferation of software. Attachment of a licensing module to a software application causes that software application to report back to a licensing server at some point. If the client module or software application is configured to report back the first time that the software application is brought up, a software provider can keep track of, or audit, which and how many machines the software application is operating on. The licensing server can be set to initially enable any request received for the software application being audited. At the discretion of the software provider, the software application can be turned off at anytime. The software provider can respond to subsequent licensing enablement or validation requests by supplying a prompt inviting the user to purchase a license. A variation on the audit function can be used to log questionable user activity.

An additional security aspect of systems in accordance with the invention is the provision of an activities log in the client module, licensing modules, or license server. Such a log gathers information associated with any or all validation requests, or irregularities. Information gathered in such a log may include, but is not limited to, data/time stamps (to ensure periodic checking is not interfered with), and the identity of a machine corresponding to the origin of an inquiry request. The foregoing information can be used to facilitate the investigation of a pirate if the pirate's activities require legal attention.

Software version control can be provided in accordance with another aspect of the invention. Software version number information can be passed upstream in the context of a license validity inquiry request message. If the software version has expired, a message can be passed back in the status field of the authorization ID indicating this information to a user. The user may then be asked to license and procure a new version of the software. Software version control can be used to prevent version collision, and to force users to stop using expired software versions by simply disabling them.

Another aspect of the invention is license system version control. By monitoring and controlling the version of the license system that is operating, a common denominator between client and agent communications can be main-

tained. Alternatively, in the event that different versions are allowed, appropriate translations of data fields can be made to ensure the different versions can still communicate. In a preferred embodiment, a given client can only communicate with an agent of the same or newer version. Licensing system version control also permits modifications to be made to improve, enhance, or entirely change an encryption schemes used by the licensing system.

The present invention also can be used to maintain a software use billing system for use with commercial client applications. A billing system in accordance with the invention can use client application and feature use information from the audit system. The audit system information can be translated into billing statements. Consequently, a commercial client can optionally be billed only for what they have used, rather than a set up-front, annual, or monthly amount.

Most computers sold today have pre-installed modems. Individual desktop machines operating in a DOS, Windows, OS/2, or like environment, can utilize their pre-installed, or aftermarket modems to initiate communications with an upstream agent or license server. Most commercial computers are designed for incorporation into network settings. Commercial computing systems, such as workstations, may operate in a UNIX environment. The UNIX environment is well suited to operation in accordance with the invention. Most UNIX workstations are furnished with network cards permitting them to be network connected. Such networked computers thus have ready direct access to Internet or gateway nodes through which a license server or licensing module can be accessed.

The particular connectivity associated with a system incorporating the invention is not critical. For example, a desktop machine can contain an Ethernet network card, ISDN connection card, Internet card, conventional modem, terminal adapter, or like device, with which to gain access to and communicate with a license server or licensing module. Any accepted form of connection between two more computers can be used. A communications protocol in an Internet environment is based on a TCP/IP protocol, or a derivative thereof. However use of other communication protocols such as IPX/SPX (NOVEL), or like protocols, also are suitable for facilitation of communication in the present invention.

When using the Internet a the medium through which to check license validity, preferred embodiments of systems according to the invention operate using a TCP (Transfer Control Protocol) mechanism rather than a UDP (Universal Datagram Protocol) mechanism. In a TCP communication, a connection is first formed before any information is transferred. That is, an interactive two-way session is established. In a UDP communication, a message is sent without forming a connection. The message is routed in accordance with its destination address information. The message's receipt is not guaranteed. More importantly, response time to a messages sent by UDP is unknown. More specifically, if heavy network traffic is present, a UDP-based message may be delayed. Consequently, there can be an unacceptable delay in enabling a client application. Using a TCP messaging format guarantees a response, even if that response indicates that a connection cannot be made. A decision can be made whether to re-attempt communication immediately or at a later time. A TCP connection also is more secure because data is not being cast into the network without guarantee of receipt, as is the case with UDP communications.

One scenario where UDP communication can be utilized is in an audit system incorporating the invention. A client application can generate and send a UDP message when it is initially brought up, or whenever a computer upon which the client application is loaded is attached to a public network.

In a non-connected or portable computer, such as a laptop, an exemplary licensing system in accordance with the invention can involve running an internal licensing module, or an agent component thereof, on the laptop computer itself. An exemplary embodiment of such an arrangement is depicted in FIG. **7**. The agent component **706** of the licensing module **702** acts on behalf of a license server **712** during the periods that the portable computer **700** is out of communication with the license server **712**. The agent component **706** can receive and respond to license validity inquiries generated by a client module **704** on software application **705** that is loaded on the portable **700** while the portable **700** is in transit. Audits of client application use can be stored in the cache **708** of the licensing module **702**. When the portable **700** is re-connected to a network access point, such as an Internet **716** gateway, the client component **710** of the licensing module **702** can establish communication with the agent module **714** in the license server **712** to refresh the information in its cache component **708** and/or supply any audit data upstream. If the user has not connected the laptop to a network access point for a prolonged period of time, the user can be prompted accordingly. This can be done pursuant to the expiration of a timer.

Systems not having network, or Internet access, can still be enabled in an automated, albeit semi-manual fashion. The licensing module in a software package to run on a non-networked machine can contain a tag indicating this fact. The user may be prompted to call a number and receive recorded information for enabling the software. This information, of course, requires initial manual entry to enable the software, and periodic manual entry to validate the software. The validation check can be performed by prompting a user to call a number to automatically receive further enablement instructions. Such instructions are only provided if the license remains valid. Optionally, instructions can be automatically be returned by fax in response to a validation request phone call.

A licensing module can be operated as a proxy agent on a firewall separating a protected computer from a public network to which the computer is attached. Such a firewall proxy agent can operate in accordance with the system disclosed in the U.S. Pat. application No. (BDSM Attorney Docket No. 025553-013), entitled: "Firewall System For Protecting Network Elements Connected To A Public Network," by Coley and Wesinger, filed on Feb. 6, 1996, and incorporated herein by reference in its entirety. The foregoing application describes a system for preventing unauthorized access to network elements protected by a firewall. The firewall operates by maintaining a plurality of proxy agents that are assigned to verify and connect any incoming access requests. A proxy agent is usually assigned based on the port number associated with an incoming access request. In the present case, however, a proxy agent for validating a license is assigned for a request originating from a desktop machine that resides behind the firewall (i.e., is protected by the firewall). Hence, the proxy agent acts as licensing module performing all of the previously described tasks associated with licensing modules. For example, a proxy agent licensing module can receive and forward a license validity inquiry request message from a client application residing on a protected computer. The proxy agent also acts on behalf of the responding agent to accept a inquiry response message and then pass the response back to the protected

computer on behalf of the responding agent. The proxy agent also can facilitate periodic updates of a cache component maintained therein. The proxy agent is the only IP address visible outside of the firewall. The foregoing system protects the identity of the client application computer because the proxy agent acts on behalf of the computer when communicating with the outside world.

In an exemplary commercial embodiment, the present invention can take the form of a software package comprised of floppy disks, a CD-ROM, or even a downloadable package. The software package may consist of a library of object modules that can be selected, as needed, by a software applications designer. The designer may select various object modules from the library for insertion into a precompiled version of a software application. The entire software application, including the selected object modules, are compiled to create a single executable client application. The selection of insertion points and frequency is left to the discretion of the designer. Validation check watchdog timers can be distributed throughout a software application. Validation checks can be inserted to correspond to various interrupts, or procedure calls within the software application (e.g., printing, saving). An opening routine in the software application can be selected for insertion of an initial license validation or enablement check.

Because the present invention involves the operation of computing systems, an exemplary embodiment of the invention can take the form of a medium for controlling such computing systems. Hence, the invention can be embodied in the form of an article of manufacture as a machine readable medium such as floppy disk, computer tape, hard drive disk, CD ROM, RAM, or any other suitable memory medium. The invention can also be embodied in a form that can be accessed and retrieved over a public network, such as the Internet, and downloaded onto a machine readable memory medium. Embodied as such, the memory medium contains computer readable program code which causes one or more computing systems upon which the licensing system is running to function or carry out processes in accordance with the present invention.

The invention has been described with respect to several exemplary embodiments. However, one skilled in the art will readily appreciate and recognize that the licensing system or method of operation in accordance with the invention can be applied in any computing system using licensed software, which systems are preferably attachable to a public network, such as the Internet. The invention provides the benefit of being able to freely distribute licensed software incorporating the invention with reduced apprehension of the software being illicitly copied or used without its being properly licensed. Alternatively, a system in accordance with the invention can be used to track and maintain records of the proliferation and use of software incorporating the invention.

The invention has been described with reference to particular embodiments. However, it will be readily apparent to those skilled in the art that it is possible to embody the invention in specific forms other than those of the embodiments described above. Embodiment of the invention in ways not specifically described may be done without departing from the spirit of the invention. Therefore, the preferred embodiments described herein are merely illustrative and should not be considered restrictive in any way. The scope of the invention is given by the appended claims, rather than by the preceding description, and all variations and equivalents which fall within the range of the claims are intended to be embraced therein.

What is claimed is:

1. A network software licensing system having self-enabling software, the network licensing system comprising:

a plurality of computers arranged in a hierarchy, the plurality of computers including client computers, server computers and agent computers, wherein the server computers are located at the highest level in the hierarchy and the client and the agent computers are located at levels in the hierarchy below the highest level;

at least one client computer having a client application loaded thereon, wherein the client application comprises a software application and a client module;

an agent computer from which at least one client computer is subtended, the agent computer having a licensing module running thereon, wherein the licensing module includes:

an agent component for communicating with the client module;

cache component for storing license records, and

a client component for communicating with an agent component;

a server computer having a license server running thereon, wherein the license server includes:

an agent module for communicating with the client component in the licensing module; and

a database for storing license records; wherein

the client component in the licensing module systematically initiates communication with the agent module in the license server over a public network to collect license records for storage in the cache component; and wherein the client module automatically initiates communication with the agent component in the licensing module at a level in the hierarchy above the client module and not the agent component in the license server to determine whether the cache component in the licensing module contains a license record corresponding to the software application.

2. The network software system claimed in claim 1, wherein a plurality of agent computers which comprise the network are connected between the client computers and the license server, any given license module being capable of communicating with any other license module and being capable of communicating with any given client module.

3. The network software system claimed in claim 1, wherein license records are organized according to class and sub-class designations.

4. The network software system claimed in claim 2, wherein an uppermost agent computer records communications from subtended agent and client computers for auditing client application use.

5. The network software system claimed in claim 4, wherein the upper-most agent computer communicates client application use data to the server computer.

6. The network software system claimed in claim 4, wherein the upper-most agent computer includes an audit tool for generating audit reports.

7. A hierarchical license system having a plurality of computers connected by a computer network, the plurality of computers arranged in a hierarchy from a lowest level to highest level and having computer software that is enabled with a license, the system comprising:

license server on a computer at the highest level in the hierarchcial license system, the license server having a database of licensing information for enabling computer software on the plurality of computers in the hierarchcial license system;

plurality of licensing modules on computers at levels in the hierarchy below the highest level in the hierarchy, the plurality of licensing modules including:

client component for communicating with agent components in licensing modules in a next level L+1 in the hierarchy,

agent component for communicating with client components in licensing modules in a previous level L−1 in the hierarchy, and

cache component for storing license information for computer software on computers in a previous level L−1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy,

wherein a client component in a licensing module at a level L in the hierarchy requests a license for enabling software from an agent component in a licensing module at a level L+1 in the hierarchy and not directly from the license server at the highest level in the hierarchy.

8. A licensing management system as claimed in claim **7** wherein the plurality of licensing modules comprises:

a library of code modules suitable for inserting into a precompiled version of the computer software, the library of code modules including a plurality of client modules.

9. The licensing management system claimed in claim **8**, wherein the library of code modules further includes:

an interupt validation generator that initiates a license validity inquiry when a piece of the computer software into which the interrupt validation generator is inserted is executed.

10. The management licensing system claimed in claim **8**, wherein the library of code modules further includes:

watchdog timer that initiates a license validity inquiry upon expiration.

11. The licensing management system claimed in claim **7** wherein a computer at the lowest level is a portable computer that includes:

a client application comprising a licensed software application and a portable client module; and

a portable licensing module comprising a portable agent component for communication with the portable client module, a portable cache component for storing license records, and a portable client component for communication with the hierarchical license server;

wherein the portable licensing module manages the portable client module to control the client application pursuant to license records maintained in the portable cache component when the portable computer is not connected to the public network, and wherein

the license module receives license record updates to its portable cache component from, and transmits client application use records to, the hierarchial license server when the portable computer is connected to the public network.

12. In a computer system having a plurality of computers connected by a computer network, the plurality of computers arranged in a hierarchy from a lowest level to highest level and having computer software that is enabled with a license, a method of enabling the computer software with a license, the method comprising the following steps:

(a) maintaining a license server on a computer at the highest level in the hierarchical license system, the license server having a database of licensing information for enabling computer software on the plurality of computers in the hierarchy;

(b) requesting a license in a licensing module at a level L in the hierarchy from a licensing module on a computer at level L+1 in the hierarchy to enable computer software on a computer at a level L in the hierarchy, the licensing modules having:

client component for communicating with agent components in licensing modules in a next level L+1 in the hierarchy,

agent component for communicating with client components in licensing modules in a previous level L−1 in the hierarchy, and

cache component for storing license information for computer software on computers in a previous level L−1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy;

(c) determining from the licensing module at the level L+1 if the requested license is in a cache component for the licensing module, and if not,

(d) sending the request for the license to a licensing module at a next higher level in the hierarchy;

(e) repeating step (d) until the requested license is located in a cache component in a licensing module at a next highest level in the hierarchy, or the highest level in the hierarchy is reached, wherein the requested license is located on the license server.

13. A computer readable medium having stored therein instructions for causing a computer to execute the method of claim **12**.

14. The method of claim **12** further comprising:

sending the located license back down the hierarchy to the licensing module which requested the license; and

updating cache components with license information in licensing modules in selected levels in the hierarchy between the level in the hierarchy at which the license was located and the level in the hierarchy at which the license was requested.

15. The method of claim **12** further comprising:

sending the located license back down the hierarchy to the licensing module which requested the license; and

updating cache components with license information in licensing modules only in a level L+1 in the hierarchy above the level L in the hierarchy from which the license was requested.

16. The method of claim **12** wherein the license server periodically sends license information to licensing modules at highest levels in the hierarchy below the license server, and the licensing modules at the highest levels in the hierarchy propagate the license information to licensing modules at lower levels in the hierarchy.

17. The method of claim **12** wherein a licensing module at a level L can only make a license request to a licensing module at a level L+1 and not directly to the license server.

18. The method of claim **12** wherein any client component in any licensing module can communicate with any agent component in any licensing module in the hierarchy, and visa-versa.

19. The method of claim **12** wherein the requesting step includes requesting the license from a client component in the licensing module at the level L in the hierarchy, and sending the request to an agent component in a licensing module at the level L+1 in the hierarchy.

20. The method of claim **12** wherein cache components for licensing modules at the lowest level in the hierarchy do not contain any license information.

21. The method of claim 12 wherein the license server organizes license information into class and sub-class designations wherein the class designations includes license information for a plurality of computers at higher levels in the hierarchy, and sub-class designations includes license information for a plurality of computers below the higher levels in the hierarchy, thereby providing efficient communications of license information between the license server, and licensing modules at all levels in the hierarchy.

22. A hierarchical computer software licensing apparatus, the apparatus comprising:

maintaining means for maintaining a license server on a computer at a highest level in a hierarchical license system, the license server having a database of licensing information for enabling computer software on a plurality of computers in the hierarchy;

requesting means for requesting a license from a computer at a level L in the hierarchy to enable computer software on the computer from a licensing module on a computer at level L+1 in the hierarchy, the licensing modules having:

client component for communicating with agent components in licensing modules in a next level L+1 in the hierarchy,

agent component for communicating with client components in licensing modules in a previous level L−1 in the hierarchy, and

cache component for storing license information for computer software on computers in a previous level L−1 in the hierarchy, wherein the cache component is updated periodically with license information from agent components in a next level L+1 in the hierarchy;

determining means for determining from the licensing module at the level L+1 if the requested license is in a cache component for the licensing module, and if not,

sending means for sending the request for the license to a licensing module at a next higher level in the hierarchy;

locating means for locating requested the license in a cache component in a licensing module at a next highest level in the hierarchy, or at the highest level in the hierarchy, wherein the requested license is located on the license server.

23. The system of claim 1 wherein the public network is the Internet.

24. The system of claim 1 wherein the communication between the client computers and the agent computers are encrypted.

25. The system of claim 1 wherein the client computers and agent computers use a shared key to initialize encryption engines contained on the client and agent computers, wherein the encryption engines operate using the shared key.

* * * * *